Type de document	Contrôle	Classe	Tle	Durée	1h25	Date	21/10/2022
Thème et contenu(s)	Structures linéaires de données, immutabilité, algorithmes gloutons et algorithmes de tri						

Tous les documents sont strictement interdits. Pas de calculatrice. Vous pouvez utiliser une fonction *même si vous n'avez pas réussi à la coder* aux questions précédentes*. Cela vous permet d'avancer* dans le contrôle sans être bloqués.

9r33Dy d3c2B1N 4nd bubbL3-50R71N'

(« greedy decToBin and bubble-sorting » en Leetspeak)

Énoncé

On se propose d'implémenter un algorithme glouton pour convertir un entier inférieur ou égal à 255 en un octet. C'est en fait l'une des deux méthodes de conversion base 10 □ base 2 (l'autre méthode étant celle des restes de la division euclidienne par deux)
Cette méthode, que nous adaptons ici à un octet seu, consiste à :

1. Choisir l'entier à convertir (inférieur ou égal 255)
2. Parcourir toutes les puissances de 2 entre 2^7 et 2^0 (l'ordre est important) et, pour chacune :
    a. Si elle peut être contenue par le nombre écrire 1 et retrancher sa valeur au nombre existant : ce résultat devient le nouveau nombre
    b. Si elle ne peut pas être contenue par ce nombre, écrire un 0 et passer à la puissance de 2 suivante (qui sera donc inférieure du fait de l'ordre)
3. Recommencer jusqu'à ce que le nombre vaille 0

En d'autres termes, on cherche toujours à enlever au nombre qu'on considère « le plus gros morceau possible », d'où le caractère glouton de cet algorithme.

Exemple avec le nombre 143 :
La valeur 2^7 = 128 est contenue par 143. J'écris **1** et je retranche 128 à 143 : 143-128 = 15. 15 est mon nouveau nombre
La valeur 2^6 = 64 ne peut pas être contenue par 15, j'écris **0**
La valeur 2^5 = 32 ne peut pas être contenue par 15, j'écris **0**
La valeur 2^4 = 16 ne peut pas être contenue par 15, j'écris **0**
La valeur 2^3 = 8 peut être contenue par 15, j'écris **1** et je retranche 8 à 15 : 15-8 = 7
7 est mon nouveau nombre
La valeur 2^2 = 4 peut être contenue par 7, j'écris **1** et je retranche 4 à 7 : 7-4 = 3
3 est mon nouveau nombre
La valeur 2^1 = 2 peut être contenue par 3, j'écris **1** et je retranche 2 à 3 : 3-2 = 1
1 est mon nouveau nombre
La valeur 2^0 = 1 peut être contenue par 1, j'écris **1** et je retranche 1 à 1 : 1-1 = 0
0 est mon nouveau nombre mais comme justement il est égal à zéro, mon algorithme se termine. En rassemblant dans l'ordre mes 0 et mes 1 j'obtiens la valeur binaire du nombre 143 : **10001111**

Rappels : Un **octet** est une suite de **8 bits**, donc en **langage binaire**. Un **bit** est la plus petite unité de mémoire en informatique et peut valoir **0** ou **1**. Par exemple, 00000000, 11111111, 00001010, 10001111 sont des octets et valent, respectivement, en base 10 : 0, 255, 10 ,143.

Questions (/21 pts)

I - a) Premières puissances de 2 (3 pts)

Écrivez la fonction `premieresPuissancesDeux(n)` qui prend en paramètre `n`, représentant les `n` premières puissances de 2, et **renvoie** une liste composée des entiers de 2^0 à 2^{n-1}
Par exemple, la commande `print(premieresPuissancesDeux(16))` doit renvoyer `[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768]`

I - b) Exemple d'appel (1 pt)

Écrivez la ligne de code qui permet d'appeler cette fonction et de placer le résultat dans une variable que vous appellerez `listePuissances`. Vous passerez l'entier 8 comme paramètre à la fonction

I - c) Tri (4 pts)

Comme vous l'avez remarqué, les valeurs sont triées par ordre croissant. Créez une fonction `triBulle(l)` qui prend une liste `l` passée en paramètre et **renvoie** une liste triée par **ordre décroissant**. Par exemple, `triBulle([9,45,2,7,11,8])` renverra `[45, 11, 9, 8, 7, 2]`

I - d) Exemple d'appel (1 pt)

Écrivez la ligne de code qui permet d'appeler cette fonction sur la liste `listePuissances` et de placer le résultat dans une variable que vous appellerez `maListeTriee`

I - e) Test logique (2 pts)

Pour cette question on considère qu'on dispose d'une variable `n` de type entier. On peut donc s'en servir sans la déclarer. Écrivez le bout de code Python (structure conditionnelle) qui affiche « bravo » si cette variable est inférieure ou égale à 255, « loupé » sinon

I - f) Concaténation (1 pt)

On dispose de la variable s suivante telle que `s = "bracadabr"`. Écrivez la ligne de code qui ajoute un `a` au début de cette chaîne. Écrivez la ligne de code qui ajoute un `a` à la fin de cette chaîne

I - g) Question de cours (1 pt)

Puis-je écrire s[1] = 'b' (ou n'importe quelle autre caractère) ? Pourquoi ? Comparer avec la concaténation

I - h) Implémentation finale (6 pts)

Créez une fonction `greeedyDec2Bin(n)` qui prend en paramètre un entier `n` et renvoie un octet sous forme de chaîne de caractères. Si n est supérieur à 255, la fonction doit renvoyer la chaîne « ERROR ». Rappel : est fortement conseillé d'utiliser toutes les questions précédentes pour vous aider, même si vous n'avez pas tout codé. Vous avez donc le droit de les appeler dans `greeedyDec2Bin(n)`

I - i) Appel final (2 pts)

Créez une variable `res` qui contiendra le résultat de l'appel de `greeedyDec2Bin(85)`. Affichez cette variable. Dites de quel type elle est. Donnez le résultat de cet appel (donc sous forme d'octet)